

①

Micro Controllers

// Assembly Commands //

// Directives //

Assembly is written as:

[label:] Mnemonic [Operands] (; comment)

→ between [] can be existed or can not.

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
ORG	ORG 500H ↓ address • indicates the beginning of the address "1st instruct. in the program" also can be used with DB directive	0		Ex: ORG 300H SQR_TABLE: DB 0, 1, 4, 9	
			END	END • Indicates the beg the assembler that this is the source file end • It is the last line in source file • Anything after it is ignored	0
DB	DATA1: DB 28 DATA2: DB 39H DATA3: DB 01101010B DATA4: DB "2591" ↳ ASCII 1 Byte = Character تخزين • بداية التخزين من العنوان الذي عليه label • used to define 8-bit data	Specified by the data size ↓ RAM but for ROM the DB has 0			

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
EQU	<p>COUNT EQU 25</p> <p>↓</p> <p>MOV R3, 25 COUNT</p> <p>في حاجة لـ COUNT</p> <p>بتتبدل 25 بـ</p> <p>(تبدل text بـ text)</p> <ul style="list-style-type: none"> Used to define a constant without occupying a memory location. 	0			
BIT	<p>LED BIT P1.7</p> <p>↓</p> <p>It assigns P1.7 to be called LED within the code.</p> <p>It is used with single bits.</p> <p>Ex BIT P2.3</p> <p>↓ ↓</p> <p>name the bit</p>	0			

①

Microntrollers

// Assembly Commands //

"Instructions"

Assembly is written as:

[label:] Mnemonic [Operands] [comment]

=> between [] can

Mnemonic	Comment	memory	Mnemonic	Comment	memory
MOV	<p>MOV dest, src</p> <ul style="list-style-type: none"> copies src into dest. 23 → value 23H → Hex. value 0F9H <ul style="list-style-type: none"> to make the assembler understand that F is not a character 	2 Bytes		<ul style="list-style-type: none"> MOV A, 7F2H <ul style="list-style-type: none"> Illegal because 7F2H needs bigger than 8 bits Affects CY "flag" 	
	<ul style="list-style-type: none"> MOV A, 5 <ul style="list-style-type: none"> $A = 5H (= 5)$ $= 05H$ $= 0000\ 0101$ it assumes the rest of a register is zeros as 5 only need 4 bits (0101) to be represented. MOV R4, R7 <ul style="list-style-type: none"> → illegal between Rn registers 		ADD	<p>ADD A, src</p> <ul style="list-style-type: none"> $A = A + src$ <ul style="list-style-type: none"> it must be A and A only Affects CY, OV, AC, P "Flags" Src could be register, address or immediate data Memory to memory arith. operations are not allowed 	1 Byte

Mnemonic	Comment	memory	Mnemonic	Comment	memory						
PUSH	<p>PUSH Address</p> <p>PUSH 6</p> <p>↓ means memory location number 06H (6) "which is R6 also"</p> <p>• It increments SP by 1, ↓ stack pointer then, puts the value within the address into the next place in the stack (which is pointed by SP after its increment)</p> <p>• SP is 07 by default at the operation</p> <p>• ex: MOV R6, #25H PUSH 6</p> <p>↓ SP = SP + 1 = 7 + 1 = 8</p> <div><table><tr><td>0A</td><td></td></tr><tr><td>09</td><td></td></tr><tr><td>08</td><td>25H</td></tr></table><p>from R6 which has the address 6 (06H)</p><p>• PUSH A → invalid → only addresses</p></div>	0A		09		08	25H	2 Bytes		<p>• Transfers the value of the current stack place pointed by SP to the address specified and decreases SP by one</p>	
0A											
09											
08	25H										
			DJNZ	<p>DJNZ reg., label</p> <p>↓ Decrement Jump if not zero</p> <p>• decrements the register by 1 then it goes to the label if the register ≠ 0</p> <p>ex: MOV R3, #10 NEXT: MOV R2, #70 AGAIN: CPL A DJNZ R2, AGAIN DJNZ R3, NEXT</p> <p>↓</p> <p>700 times loop</p> <p>• Label must be within; -128 + PC + 127</p>	Short Jump ↓ 2 Bytes						
POP	<p>POP Address</p> <p>POP 7</p>	2 Bytes									

(3)

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
JZ	JZ label Jump if zero	Short Jump ↓ 2 Bytes	JNB	JNB bit, label • jump if this bit $\neq 1$	Short Jump ↓ 2 Bytes
	• Jumps to label when $A = 0$		SJMP	SJMP Label Short Jump	2 Bytes
JNC	JNC label Jump if no carry	Short Jump ↓ 2 Bytes		• first byte is opcode • second byte is relative 8-bit target address ($-128 + PC + 127$) in this range	
CJNE	CJNE A, byte, label Conditional Jump if not equal	Short Jump ↓ 2 Bytes		• This is unconditional jump instruction	
	• Jump if $A \neq$ Byte		LJMP	LJMP Label Long Jump	3 bytes
	CJNE reg, #data, HERE			• First byte is opcode • second and third bytes are 16-bit target address "to any 11 place $-(2^{15}) + PC + (2^{15} - 1)$	
JC	JC label Jump if carry	Short Jump ↓ 2 Bytes			
	• Jump if $CY = 1$				
JB	JB (bit), Label • jump if this bit = 1	Short Jump ↓ 2 Bytes			

Mnemonic	Comment	memory	Mnemonic	Comment	memory
ACALL	<p>ACALL Label</p> <p>Absolute Call</p> <ul style="list-style-type: none"> 11 bits are used for address within 2k-byte range when a subroutine is called, processor saves on the stack the address of the instruction immediately below the call instruction 	2 Bytes	RET	<p>RET</p> <ul style="list-style-type: none"> Every subroutine needs RET as the last instruction to transfer control back to the caller pops the return address from the stack into the PC and resumes within the instructions after the call instruction <p>ex:</p> <pre> _____ _____ LCALL AGAIN _____ _____ ORG 300H _____ AGAIN: DJNZ R5, AGAIN RET </pre> <p>ex: → page 116 117</p>	1 Byte
LCALL	<p>LCALL Label</p> <p>Long Call</p> <ul style="list-style-type: none"> First byte is the opcode Second and third bytes are used for address of target subroutine which is located anywhere within 64k byte address space 	3 Bytes	NOP	<p>NOP</p> <ul style="list-style-type: none"> no opcode Does nothing No registers or flags are affected used widely for delays 1 machine cycle 	1 Byte

5

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
CPL	CPL bit ↓ complement • ex: CPL P1, 2 • $P1, 2 = (P1, 2)'$	1 Bytes		• The data is Anded with \square and the result is returned back to the latch in order to affect the external pin ex: ANL P1, A → $P1 = P1$ AND A • $\text{ANL P1, } \text{XSH}$ * Read-Modify-Write	
SETB	SETB bit ↓ Set bit • ex: SETB P1, 2 • $P1, 2 = 1$	1 Bytes			
CLR	CLR bit ↓ clear • ex: CLR P1, 2 • $P1, 2 = 0$	1 Byte			
JBC	JBC bit, Label ↓ Jump if bit = 1 and clear • Jumps to label if bit = 1 then clears the bit after that.	2 Bytes "Short" Jump	ORL	ORL Δ, \square ↓ OR Latch $\Delta = \Delta \text{ OR } \square$ ↓ part * Read-Modify-Write	2 Bytes
			XRL	XRL Δ, \square ↓ XOR Latch $\Delta = \Delta \text{ XOR } \square$ ↓ part * Read-Modify-Write	2 Bytes
ANL	$\text{ANL } \Delta, \square$ ↓ AND Latch • It reads internal latch of Δ "part" then brings data to CPU	2 Bytes			

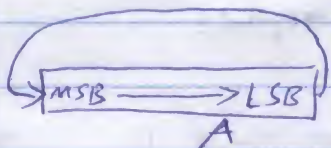
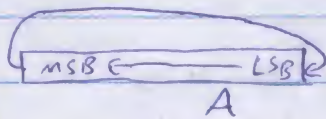
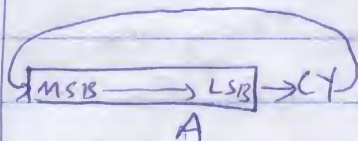
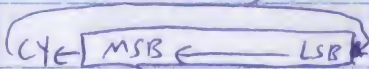
6

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
MOVC	MOVC A, @A+DPTR ↓ Move code • used to load data from the ROM "look-up table" • A = data which was in ROM at the location @A+DPTR ↓ points to some location	1 Byte	DEC	DEC @R1 ↓ Decrement a register or a memory location by 1	1 Byte
			ADDC	ADDC A, #3H ↓ Add with carry • $A = A + 3H + C$ ↓ C ↓ carry	1 Byte
MOVL	MOVL @R0, A • used to transfer data between A and locations in external memory "when used"	1 Byte	DA	DA A ↓ Decimal Adjust this is A only • It corrects the problem of BCD addition (adds 6 to A) to adjust 2 BCD numbers addition. • Works only after ADD instruction but not after INC • ex: ADD A, B DA A	1 Byte
INC	INC A • Increments a register or a memory location by 1	1 Byte			

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
	<ul style="list-style-type: none"> After ADD or ADDC instruction ① if the lower nibble is greater than 9 or [if $AC=1$], it adds 0110 (6) to the lower nibble ② if the upper nibble > 9 or [if $CY=1$], it adds 0110 (6) to the upper upper nibble 		MUL	MUL AB <ul style="list-style-type: none"> $BA = A * B$ It performs byte by byte multiplication operation and it will result in a 16-bit result. The result will be in the form: B = high byte of result A = Low byte of result Bytes are assumed to be unsigned (A, B) ex: $MOV A, \#25H$ $MOV B, \#65H$ $MUL AB$ 	1 Bytes
SUBB	SUBB A, SRC Subtract with borrow <ul style="list-style-type: none"> $A = A - SRC - CY$ There is no SUB to make only SUB operation, we have to make $CY=0$ prior to instruction execution It takes 2's complement of SRC then adds it to A then complements CY 	1 Byte			
			DIV	DIV AB <ul style="list-style-type: none"> $A = A / B$ $(B = A \% B)$ ↳ remainder if $B=0$ (division by zero) → $OV=1$ → indicating error 	1 Byte

mnemonic	Comment	memory	mnemonic	Comment	memory
	<ul style="list-style-type: none">Byte by Byte divisionBytes are assumed to be unsigned (A, B)		XRL (2)	XRL dest, src XOR Logic A register address immediate • Can be used with bits	2 Bytes
ANL (2)	ANL dest, src AND Logic performs Logic AND on src, dest. • src dest, is normally A • src can be register, address or immediate • Can be used with bits	2 Bytes	CPL (2)	CPL A complement complements A	1 Byte
			CJNE (2)	CJNE src, dest CJNE dest, src, Label A or Rn register register address immediate • the src, dest. remain the same and do not change • It changes CY flag dest > src → CY = 0 dest < src → CY = 1 • It is a subtraction operation in reality except the operands are unchanged	Short Jump 1 2 Bytes
ORL (2)	ORL dest, src OR Logic • dest, is normally A. • src can be register, address or immediate • Can be used with bits	2 Bytes			

9

Mnemonic	Comment	Memory	Mnemonic	Comment	Memory
RR	RR A Rotate Right	1 Bytes	SWAP	SWAP A ✓ must be A Swaps lower nibble of A with the higher nibble of A	1 Byte
					
RL	RL A Rotate Left	1 Bytes	RETI	RETI The last instruction at an Interrupt Service Routine (ISR) which returns back to the main program. It clears the interrupt-in-service flag also, it clears TCON.1, TCON.3 and it also indicates that the interrupt pin can accept another interrupt, RET can not do that.	1 Bytes
					
RRC	RRC A Rotate Right with carry	1 Byte			
					
RLC	RLC A Rotate Left with carry	1 Byte			
					
Mov (2)	Mov P2.1, C Can be used with bits also	2 Bytes			

Notes: • Instructions that operates on single bits is called "single-bit instructions"

• Look at the examples

• Memory field specified is about the size that an instruction has within the ROM

• An Instruction which use 2 Bytes from ROM is called "2-Byte instruction" and so on.

• Directives do not consume any size of ROM, they are for the assembler also to make understand some notations or notes in the code.